Analysis of Banker’s Algorithm

Let m be the totla number of resource types

Let n be the total number of customers

Consider this snippet of code from the Banker’s Algorithm

if (finish(Ci) == **false** && temp\_need(Ci) <= work) {

possible = **true**;

work += temp\_allocation(Ci);

finish(Ci) = **true**;

}

Temp\_need and work are 1 dimensional arrays of size m. Thus the comparison of temp\_need and work [temp\_need(Ci) <= work ] and updating of the work array [work += temp\_allocation(Ci)] each have complexity m.

Thus, this if loop has complexity O(m+m) = O(2m) = O(m)

This if loop is nested within a for loop

for(customer Ci = **1**:n) {

if (finish(Ci) == **false** && temp\_need(Ci) <= work) {

possible = **true**;

work += temp\_allocation(Ci);

finish(Ci) = **true**;

}

}

The for loop iterates through all customers [(customer Ci = **1**:n)], hence it would iterate n times. Thus the for loop would have a complexity of O(n\*m).

This for loop is in turn nested in a while loop

while(possible) {

possible = **false**;

for(customer Ci = **1**:n) {

if (finish(Ci) == **false** && temp\_need(Ci) <= work) {

possible = **true**;

work += temp\_allocation(Ci);

finish(Ci) = **true**;

}

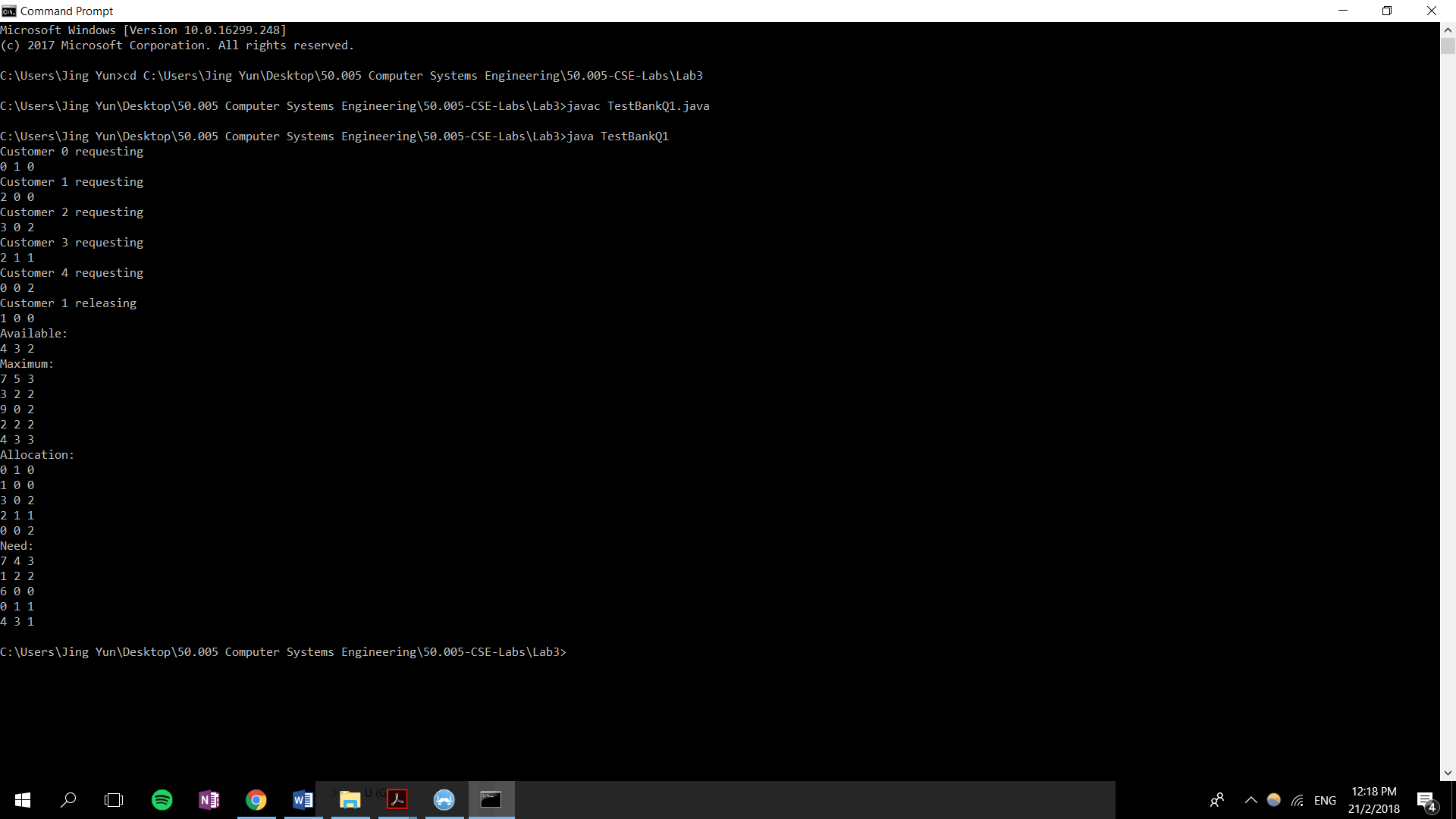
}

}

In the worst case the while loop would iterate n times.

Thus, the complexity of Banker’s Algorithm is **O(m\*n\*n)**

Test Bank Q1



TestBankQ2

